Simplified stage-based modeling of multi-stage stochastic programming problems

Ronald Hochreiter
Department of Statistics and Decision Support Systems, University of Vienna

11th International Conference on Stochastic Programming (SPXI).
Vienna, Austria. August 2007.
Overview

- Introduction: Multi-stage decision process optimization
- Multi-stage stochastic programming - Main issues
- Modeling multi-stage stochastic programming problems
- Simplified stage-based multi-stage modeling
- Modeling Examples
- Conclusion
Multi-stage Stochastic Decision Process

Discrete-time decision processes considered:

- **Sequence of decisions.** At each decision stage $t = 0, \ldots, T$ do:
  - Observe the realization of random variable $\xi_t$.
  - Take a decision $x_t$ based on all observed values $\xi_0, \ldots, \xi_t$.

- **At stage $T$.** Sequence of decisions $x = (x_0, \ldots, x_T)$ with realizations $\xi = (\xi_0, \ldots, \xi_T)$ leads to cost $f(x, \xi)$.

- **Goal.** Find a sequence of decisions $x(\xi)$, which minimizes a functional (commonly the expectation) of the cost $f(x(\xi), \xi)$.

**Multi-stage:** At least one intermediary stage between root and terminal stage.
Multi-stage Stochastic Programming

\[
\begin{align*}
\text{minimize } & \quad x : \quad F \left( f \left( x(\xi), \xi \right) \right) \\
\text{subject to } & \quad (x(\xi), \xi) \in \mathcal{X} \\
& \quad x \in \mathcal{N}
\end{align*}
\]

- Multi-variate, multi-stage stochastic process $\xi$.
- Constraint-Set $\mathcal{X}$ defining feasible $(x, \xi)$.
- Set $\mathcal{N}$ of functions $\xi \mapsto x$, such that $x_t$ is based on realizations up to stage $t (\xi_0, \ldots, \xi_t)$ only (non-anticipativity constraints).

Remark. The scenario tree approximation of the underlying stochastic process will inherently fulfill the non-anticipativity constraints.
Multi-stage stochastic programming - Main issues

**Issue One** - Modeling underlying decision problem - Multi-stage models and scenario model (tree) handling are considered to be too complex to be used in companies for real-world applications. Communication of tree-based models to non-experts is complicated.

**Issue Two** - Modeling underlying uncertainty - A discrete tree approximation of the underlying stochastic process has to be generated in order to numerically compute a solution. The quality of the scenario model severely affects the quality of the solution (garbage in → garbage out).

Both issues are valid since the inception of stochastic programming. However, are they properly solved?
Modeling multi-stage stochastic programming problems (1)

Most stochastic programming modeling environments summarized in:


Some recent developments were reported by:

- Lopes, L. - PhD, Northwestern 2003
- van Delft, Ch. and Vial, J.-P. - *Automatica* 2004
- Fourer, R. and Lopes, L. - *Optimization Online* 2006
Modeling multi-stage stochastic programming problems (2)

**Design philosophy.** Complete decoupling of scenario tree modeling and handling from the decision problem modeling process. Three-layered approach: explicit decoupling of modeling and (scenario) tree handling.

1. **Decision problem layer.** Decision problem modeler only concerned with actions/decisions at stages.

2. **Scenario tree layer.** Creating a scenario tree which optimally represents the subjective beliefs of the decision taker at each node.

3. **Data layer.** Data structures, how to (memory-)optimally store large trees, and access ancestor tree nodes fast, . . .

**Design goal.** Focus on usability, and model readability.
Three-layered approach (1)
Three layered approach (2)

Scenario Tree Handling. Need for a coherent interface to handle scenario trees. Still no common standard for representing discretized stochastic processes available. Lack of commercial interest?!

Node-based vector/matrix data format of a scenario tree:

\[ V(n, d) \]  
\[ A(n) \]  
\[ P(n) \]  
\[ Z(n) \]  
\[ T(n) \]

\[ d \]-dimensional value of node \( n \)
ancestor node of node \( n \)
probability to reach node \( n \) from its ancestor
probability of scenario terminating at node \( n \)
stage of node \( n \)

Underlying concept. Node-sets \( N_t \), include all tree nodes of stage \( t \). Deterministic root-node and root-stage indexed with 0.
A simple multi-stage model

Stylized multi-stage stochastic programming example from (Heitsch et al., 2006):

- Optimal purchase over time under cost uncertainty,
- Uncertain prices are given by $\sigma_t$.
- Decisions $x_t$: amounts to be purchased at each time period $t$.
- Minimize expected costs such that prescribed amount $a$ is achieved at $T$.

\[
\begin{align*}
\text{minimize} & \quad \mathbb{E} \left( \sum_{t=1}^{T} \xi_t x_t \right) \\
\text{subject to} & \quad s_t - s_{t-1} = x_t \forall t = 2, \ldots, T \\
& \quad s_1 = 0, s_T = a, x_t \geq 0, s_t \geq 0,
\end{align*}
\]

where $s_t$ is a state variable containing the amount at time $t$. 
A simple multi-stage model MusMod - formulation

deterministic a: T;
stochastic V, x, s: 0..T;
stochastic nonAnticitpativity: 1..T;
stochastic constraintRootStage: 0;
stochastic constraintTerminalStage: T;

param a, V;
var x >= 0, s >= 0;

maximize objFunc: E(V * x, 0..T);
subject to nonAnticitpativity: s - s(-1) = x;
subject to constraintRootStage: s = 0;
subject to constraintTerminalStage: s = a;
Additional keywords (for parameters, variables, and constraints)

- deterministic `variable-name: stage-set;`

- stochastic `variable-name: stage-set;`

- Stochastic parameters are defined on the underlying tree `node` structure.
- Deterministic parameters are defined on the `stage` structure, i.e. same value for all nodes in the respective stage.

Remark. Stage-sets may be single stages, ranges, or lists.
MusMod - AMPL extension - direct modeling changes

Additional functions:

- stochastic variables (recourse)
  \[ \text{variable-name}(\text{recourse-depth}, \text{parameters}) \].
  Recourse-depth equals number of stages, commonly -1.

- expectation \[ E(\text{stochastic-variable-name}, \text{stage-set}) \].

Possible extensions:

- quantiles \[ Q(\text{stochastic-variable-name}, \text{stage}, \alpha) \].

- probabilistic constraints
  \[ P(\text{stochastic-variable-name}, \text{stage}, \leq, \alpha) \].
MusMod - Stage-set parsing, node-set creation

Node-sets parsing & creation.

• Add one stage-set for the whole horizon (0..T).

• Parse all stage-sets \( \mathcal{T} \) defined
  
  – directly with keywords stochastic and deterministic, and
  
  – within the objective special function \( E() \).

• For each stage-set - given one specific scenario tree - create appropriate node-sets containing all nodes of the respective stages.
Example: Simple three-stage ($t = 0,1,2$) binary tree, (uni-variate) starting value: 10. Up 1 with $p = 0.6$ and down 1 with $p = 0.4$, i.e.

\[
\begin{array}{ccccccc}
  n & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\
  V[n] & 10 & 11 & 9 & 12 & 10 & 10 & 8 \\
  A[n] & 0 & 0 & 1 & 1 & 2 & 2 & 2 \\
  T[n] & 1 & 1 & 2 & 2 & 2 & 2 & 2 \\
  P[n] & 1 & 0.6 & 0.4 & 0.6 & 0.4 & 0.6 & 0.4 \\
  Z[n] & 1 & 0.6 & 0.4 & 0.36 & 0.24 & 0.24 & 0.16 \\
\end{array}
\]

Node- and stage-sets: Using the above inventory example:

<table>
<thead>
<tr>
<th>Model</th>
<th>Node-Set</th>
<th>Stages</th>
<th>Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0..T)</td>
<td>0</td>
<td>0 1 2</td>
<td>0 1 2 3 4 5 6</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>T</td>
<td>2</td>
<td>2</td>
<td>3 4 5 6</td>
</tr>
<tr>
<td>1..T</td>
<td>3</td>
<td>1 2</td>
<td>1 2 3 4 5 6</td>
</tr>
</tbody>
</table>
MusMod - Conversion example - Variables and Parameters

1. Replace stochastic parameters and variables by a node-set definition, and

2. replace deterministic parameters and variables by stage-set definitions.

deterministic a: T;
stochastic V, x, s: 0..T;
param a, V;
var x >= 0, s >= 0;

param a[stageSet2], V[nodeSet0];
var x[nodeSet0] >= 0, s[nodeSet0] >= 0;
For each stochastic constraint add as many deterministic equivalent constraints as nodes in the respective node set, i.e.

```
stochastic constraintRootStage: 0;
subject to constraintRootStage: s = 0;
```

```
subject to constraintRootStage: s[0] = 0;
```
Deterministic parameters in stochastic constraints make use of the stage mapping information $T[n]$:

```plaintext
stochastic constraintTerminalStage: $T$;
subject to constraintTerminalStage: $s = a$;

subject to constraintRootStage: $s[3] = a[T[3]]$;
subject to constraintRootStage: $s[4] = a[T[4]]$;
subject to constraintRootStage: $s[5] = a[T[5]]$;
subject to constraintRootStage: $s[6] = a[T[6]]$;
```
Recourse constraints make use of the ancestor information $A[n]$. Higher depths are integrated recursively.

\[
\text{stochastic nonAnticitpativity: } 1..T; \\
\text{subject to nonAnticitpativity: } s - s(-1) = x; \\
\text{subject to nonAnticitpativity: } s[1] - s[A[1]] = x[1]; \\
\text{subject to nonAnticitpativity: } s[2] - s[A[2]] = x[2]; \\
\text{...} \\
\text{subject to nonAnticitpativity: } s[6] - s[A[6]] = x[6];
\]

**Further advantage:** No explicit tree formulation in the model anymore.
MusMod - Conversion example - Objective function

Objective function replacements based replacing \( E() \) by sums using the stage probabilities \( Z[n] \):

\[
\text{maximize objFunc: } E(V \ast x, 0..T);
\]

\[
\text{maximize objFunc: } \\
\quad \left( \sum_{n \in \text{nodeSet0}} Z[n] \ast (V[n] \ast x[n]) \right);
\]
Multi-stage stochastic Asset Liability Management

maximize \(x\) \[\sum_{n \in \mathcal{N}(T)} Z(n)w_n + \kappa(\gamma - \sum_{n \in \mathcal{N}(T)} \frac{Z(n)z_n}{1-\alpha})\]

subject to

\[\sum_{a \in \mathcal{A}} x_{n,a} \leq \beta = w_n\quad \forall n \in \mathcal{N}(0)\]

\[\forall a \in \mathcal{A} : x_{n,a} \leq V(n, a)x_{A(n),a} + b_{n,a} - s_{n,a}\quad \forall n \in \mathcal{N}(1..T-1)\]

\[\sum_{a \in \mathcal{A}} b_{n,a} \leq \sum_{a \in \mathcal{A}} s_{n,a}\quad \forall n \in \mathcal{N}(1..T-1)\]

\[w_n = \sum_{a \in \mathcal{A}} x_{n,a} + f_S(n)\quad \forall n \in \mathcal{N}(T)\]

\[w_n = (\sum_{a \in \mathcal{A}} V(n, a)x_{A(n),a}) + f_S(n)\quad \forall n \in \mathcal{N}(T)\]

\[z_n \geq \gamma - w_n\quad \forall n \in \mathcal{N}(T)\]

\(x_{n,a}\) amount of money held in each asset \(a\)

\(b_{n,a}, s_{n,a}\) amount bought and sold

\(w_n\) current wealth

\(f_t\) (deterministic) liabilities

\(\beta\) initial budget

\(\kappa\) risk aversion parameter

\(\alpha\) AVaR quantile level

\(z_n, \gamma\) auxiliary variables (AVaR)
maximize \( \mathbb{E}(w_T) + \kappa(\gamma - \mathbb{E}(\frac{z_t}{1-\alpha})) \)

subject to

\[
\sum_{a \in A} x_a \leq \beta = w \quad (t = 0)
\]

\[
\forall a \in A : x_a \leq V_a x_a^{(-1)} + b_a - s_a \quad (t = 1, \ldots, T - 1)
\]

\[
\sum_{a \in A} b_a \leq \sum_{a \in A} s_a \quad (t = 1, \ldots, T - 1)
\]

\[
w = \sum_{a \in A} x_a + f \quad (t = 1, \ldots, T - 1)
\]

\[
w = \sum_{a \in A} V_a x_a^{(-1)} + f \quad (t = T)
\]

\[
z \geq \gamma - w \quad (t = T)
\]
ALM example - Simplified notation (AMPL extension, 1)

param alpha; param beta; param kappa;
param assets; set ASSET := 1 .. assets;
param V{ASSET};

var x{ASSET} >= 0, b{ASSET} >= 0, s{ASSET} >= 0, w >= 0;
var f; var g; var z >= 0;

maximize objFunc: E(wealth, T) + kappa * ( g - ( E(z / ( 1 - alpha ), T) ) );

subject to cInitBudget: ( sum{a in ASSET} x[a] ) <= beta;
subject to cInitWealth: ( sum{a in ASSET} x[a] ) == w;
subject to cTradeStages{a in ASSET}: x[a] <= ( V[a] * x(-1, a) ) + b[a] - s[a];
subject to cBuySell: ( sum{a in ASSET} b[a] ) <= ( sum{a in ASSET} s[a] );
subject to cNodeWealth: w <= ( sum{a in ASSET} x[a] ) + f;
subject to cFinalStageWealth: w <= ( sum{a in ASSET} V[a] * x(-1, a) ) + f;
subject to cAVaR: z >= g - w;
ALM example - Simplified notation (AMPL extension, 2)

deterministic f: 1..T;

stochastic cInitBudget, cInitWealth: 0;
stochastic x, w: 0..T;
stochastic b, s: 1..T-1;
stochastic V, cTradeStages, cBuySell, cNodeWealth: 1..T;
stochastic z, cAVaR, cFinalStageWealth: T;
MusMod - Workflow

Model / Tree consistency check - Examples:

- Number of stages smaller than highest recourse-depth.
- Equal/Odd number of stages required by model.
Conclusion

• Multi-stage modeling completely decoupled from the scenario tree handling (stage-based modeling view).

• AMPL extension implemented as Web application.

• Framework for teaching and selling multi-stage models.

• Guideline for designing stochastic programming (XML) formats.

Contact & More Information

Web  http://www.compmath.net/ronald.hochreiter/
Email  ronald.hochreiter@compmath.net